Research on Optimization and Security Management of Database Access Technology in the Era of Big Data

Yiru Zhang

Consumer Electronics Technology, Amazon, New York, NY 10044, USA
yiruz315@163.com

Abstract: *In the era of big data, the database connection access technology is faced with huge performance and security challenges. The explosive growth of data volume and high concurrency demand in big data systems have gradually exposed problems in the performance and security of traditional database connection modes and access mechanisms. Traditional database connection modes cannot meet the needs of modern applications, exposing deficiencies such as resource consumption, response delay and connection bottleneck. In order to improve the efficiency of database access, relevant optimization techniques have been widely used to effectively alleviate the problem of data consistency in highly concurrent access and distributed databases. With the diversification of database access scenarios, data security also faces more complex challenges. Based on the current situation of database connection and access technology, this paper analyzes the bottlenecks and shortcomings reflected in the big data scenario, discusses various optimization methods and strategies, uses the relevant theoretical knowledge of database access security management, and puts forward a series of security management measures to ensure the security and stability of the database in the complex and changeable network environment.*

Keywords: *Big Data, Connectivity Access Technology, Database Connectivity, Security Management*

1. Introduction

With the rapid development of information technology, the scale and complexity of database system have reached an unprecedented height. Big data technology is widely used, including the Internet, finance, healthcare, retail and many other fields. As the core of big data system, database carries the storage, query and processing requirements of massive data, and its performance and security directly affect the efficient operation of the whole system. Under this background, the importance of database connection access technology becomes more and more prominent. The efficiency of database connection access determines the response speed of data query, and with the increase of business volume, optimizing database connection access technology under high concurrency environment has become the core challenge.

The traditional database connection management model is powerless in the face of the high concurrency demand of big data systems. When a large number of users access the database at the same time, frequent establishment and disconnection may waste system resources, increase server load, and degrade system performance. Optimization methods such as connection pool technology, distributed database architecture and cache mechanism have become the mainstream technologies to solve this problem. However, with the development of distributed architecture and cloud computing, the complexity of database connection access is also increasing, and data distribution, load balancing, transaction consistency and other problems become new problems. Database security is also facing severe challenges, especially in the multi-user, multi-application concurrent access environment, data leakage, SQL injection, permission abuse and other security risks rise. While optimizing database access performance, ensuring database security has become a common concern of academia and industry.

2. Current Status of Database Connection Access Technology

2.1 The Development of Database Connection Access Technology

The evolution of database connection access technology is closely accompanied by the development of database technology itself. In the 1960s, with the advent of Hierarchical databases and Network databases, Database connectivity used proprietary low-level interfaces, and data access required specialized interface definitions and fixed communication protocols. Although this approach can handle certain concurrency and data access requirements, it can not meet the requirements of flexibility and scalability of modern scenarios.[1]

In the 1970s, the introduction of the Relational Database model and the popularization of the Structured Query Language (SQL) revolutionized database connection access technology. Relational database adopts standardized SQL query language, which makes data access more uniform. Database access is mainly carried out through the client-server (C/S) mode, the Client communicates with the Server through SQL statements, and the server returns the query result according to the SQL request. The advantages of this mode are simple to use and strong versatility, but it is easy to encounter performance bottleneck in high concurrency environment.

Entering the 1990s, the popularity of the Internet led to a surge in demand for database access. To cope with the increasing number of concurrent requests, Persistent Connection and Short-lived Connection have become two common connection management patterns. Persistent connections reduce the overhead of frequent connection establishment by keeping database connections open between multiple requests, and are suitable for long sessions or persistent query operations. Short connections close the connection immediately after each request is completed, which is suitable for short and fast query scenarios. Connection Pooling technology is also applied to large-scale concurrent systems. By establishing and maintaining a set of database connections in advance for application reuse, the efficiency of database connection management is greatly improved.[2]

Since the 21st century, with the rise of distributed database and NoSQL database, database connection access technology has been further developed. The distributed database distributes data and access load across multiple nodes by means of Horizontal Scaling and Sharding, which significantly improves the concurrent processing capability and fault tolerance of the database. In these systems, the connection management of database access is no longer limited to a single server, but requires scheduling and load balancing across multiple nodes, and Distributed Transaction and Multi-master Replication become common connection management strategies.

2.2 The Bottleneck and Insufficiency of Database Connection

The traditional database connection technology faces many bottlenecks and deficiencies in the big data environment. The first is the high concurrency access bottleneck. With the surge in the number of users in the big data scenario, the database needs to handle a large number of concurrent requests. The traditional database connection mechanism tends to produce performance bottleneck in high concurrency scenarios. Each database connection consumes certain system resources, such as memory and CPU. When the number of concurrent connections reaches a certain threshold, server resources are exhausted, the system response time may be increased or connection timeout may occur, which seriously affects user experience. The traditional centralized database architecture also has scalability bottlenecks, and it is difficult to efficiently handle concurrent access requests from multiple clients. The second is the high connection cost. Every time a database connection is established, it involves three handshakes, authentication, resource allocation and other operations. [3]These processes consume a lot of time and system resources. In frequent short connection scenarios, the above process needs to be repeated for each request, resulting in excessive connection overhead. Although the use of connection pooling technology can alleviate this problem, the management and scheduling of connection pools still face the risk of resource exhaustion during a large number of concurrent requests, especially if the connection pool is improperly configured, which can lead to connection leakage or deadlock problems. There is also the complexity of distributed environments, where databases are deployed using a distributed architecture in big data scenarios. Distributed database faces more complexity in data storage, query and transaction processing. Although Sharding and replication mechanisms improve the scalability and availability of the system, they also increase the complexity of data synchronization and connection management between nodes. The network delay, data consistency maintenance and connection scheduling between different nodes will affect the overall performance of the system. Also, distributed transaction processing is more complex than stand-alone transaction processing, increasing the management and maintenance costs of the connection. The last is transaction processing and consistency problem, the traditional database transaction processing mechanism (ACID) in the big data scenario to ensure data consistency at the same time, it is difficult to meet the requirements of high concurrency and low latency. In particular, distributed systems increase the complexity of connection management to ensure data consistency and transaction isolation among distributed nodes. When end users access the database, they may encounter data inconsistency, which affects the execution of business logic.

3. Optimization Strategy of Database Connection Access Technology in Big Data Scenario

3.1 Connection Pool Technique

As a mature optimization method, connection pool technology can effectively alleviate the pressure brought by high concurrent access to the database. The core idea of connection pooling is that a certain number of database connections are pre-created and stored in the pool, and applications do not need to recreate new connections when they need to connect, but directly obtain existing connections from the connection pool, and then return the connections to the pool after use. This avoids the resource overhead of frequently establishing and closing connections, thereby improving the concurrent processing capability and overall performance of the system.[4]

Connection pooling technology can be optimized in several ways. One is to reduce the overhead of connection establishment and destruction, the traditional database connection needs to establish a new connection on each request, and then close the request after completion. The whole process consumes a lot of system resources, especially in frequent short connection access scenarios. This approach not only wastes time, but also increases the load on the database server. The connections allocated in advance through the connection pool can be reused multiple times, avoiding the overhead of repeated connection establishment and improving the efficiency of database access. The second is the maximum number of concurrent connection control, if the database connection is established without restriction in a high concurrency environment, it will cause the exhaustion of database server resources or even collapse. The maximum number of connections set by the connection pool limits the number of connections that can be established at a time to ensure system stability in high-concurrency scenarios. The third is connection idle management and timeout reclamation. To prevent resource waste, you can set the idle time and timeout time of the connection pool to periodically reclaim the long-term unused connections. Idle connections occupy system resources. You can use a proper timeout mechanism to close idle connections to release system resources and keep the connections in the connection pool working properly. The fourth is the distributed management of load balancing and connection pool. In a distributed database environment, connection pool not only manages the connection of a single database node, but also manages the load balancing of multiple nodes. The connection pool needs to evenly distribute requests to multiple database nodes through proper load balancing to prevent overload on a single node.

3.2 The Cache Mechanism of Database Access

The cache mechanism of database access is an important means to improve system performance and reduce database load in big data scenarios. Frequent database requests can put a lot of strain on the database server. Through the cache mechanism, the database can effectively reduce the processing burden and improve the access efficiency. The cache mechanism is usually divided into three categories: client cache, middle tier cache and database memory cache. These three kinds of caching mechanisms are combined in practical applications to achieve the best performance optimization effect.[5]

Client caching means storing query results locally on the client. For those data requests with high repetition rate and less stringent timeliness requirements, the client can reduce the number of requests to the server through caching, reducing network overhead and database pressure. Mid-tier caching is the caching layer introduced between the application server and the database. Common mid-tier caching technologies include in-memory caching systems such as Redis and Memcached. Mid-tier cache stores frequently accessed hotspot data, greatly reducing database I/O operations and improving system response speed. It is suitable for handling read-intensive operations. By caching frequently accessed data in memory, the system can read data faster and reduce the frequency of database connections. Database memory cache is a cache mechanism in database system.Modern relational and NoSQL databases use memory to cache frequently used data pages, speeding up the data query process. Database memory cache loads the most frequently accessed data into the memory, reducing disk read and write operations and improving overall query performance. This caching mechanism increases the throughput of the database while handling frequent read operations and reduces the dependence on disk I/O.

4. Database Access Security Management

4.1 Authentication and Permission Control

In database access security management, authentication and permission control are the important cornerstones to ensure data security and system integrity. Simple authentication methods are not enough to deal with diversified security threats due to the expansion of data scale and system complexity. Authentication mechanisms and permission control policies are combined with modern information security technologies to prevent unauthorized access, data breaches and potential malicious attacks through tighter management.

Authentication is an important means to confirm the validity of user identity. Traditional user name and password authentication methods are not reliable in the face of complex security environments. To increase the strength of Authentication, Multi-Factor authentication (MFA) has become a mainstream choice. The MFA combines several validation factors. Even if the attacker obtains part of the authentication information, the system security cannot be breached. Digital certificate verification using public key infrastructure (PKI) technology has also become one of the important ways to ensure the security of identity authentication. By pairing public and private keys, the server and client can realize bidirectional authentication and enhance communication security.

Permission control is a key mechanism that upgrades on the basis of authentication and allows users to access only the data and resources they are authorized to manipulate. Permission Control Through Role-Based Access Control (RBAC). RBAC simplifies rights management by assigning users to specific roles and then assigning a set of rights based on the roles. This model is not only easy to scale, but also ensures that different roles have the Principle of Least Privilege, reducing the risk of sensitive data exposure. RBAC can also be further refined into attribute access control, dynamically assigning permissions based on user, environment, and resource attributes, enhancing the flexibility and refinement of the system.

4.2 Data Encryption

Data encryption is an important part of database access security management, which aims to prevent data from being read or tampered with by unauthorized users during data transmission and storage. In the era of big data, the amount of data increases rapidly and distributed database architecture is popularized, and data encryption technology is more and more important. When dealing with sensitive data, encryption becomes a key line of defense against data breaches.

Data encryption is divided into two parts: transmission layer encryption and storage layer encryption. Transport layer encryption enables attackers to obtain sensitive information by eavesdropping on network traffic during data transmission between database clients and servers. Transport layer encryption encrypts packets by using protocols such as SSL/TLS. The purpose of encryption is that even if an attacker intercepts the data, the contents of the data cannot be read directly, and only the party with the correct key can decrypt the information. SSL/TLS not only encrypts data, but also ensures the integrity of data transmission and authentication of both parties communicating, preventing man-in-the-middle attacks. Storage layer encryption refers to the database storage layer encryption, that is, to encrypt static data. Full Disk Encryption (FDE) is a common way to protect an entire database or file system. With full encryption, administrators can ensure that even if a disk is stolen or physical media is lost, an attacker cannot read the contents of the data directly. There are also table level encryption and field level encryption, field level encryption for specific sensitive data fields, with finer control granularity. There is also key management, the core of his encryption lies in the security management of the key.[6]

4.3 Database Firewall

The Database Firewall is an important protection measure in database access security management to protect the database from external and internal malicious attacks, abnormal access behaviors, and data leaks. Database firewalls enhance the overall security of a database by monitoring, analyzing, and filtering all requests to access the database, identifying potential security threats, and preventing malicious behavior.

The firewall faces the following threats: SQL Injection is a type in which an attacker manipulates a database to read or modify unauthorized data by submitting maliciously constructed SQL statements to an application. The database firewall can resist this attack by detecting abnormal patterns, characters, and behaviors in SQL queries to prevent the execution of malicious SQL statements. The firewall filters out SQL statements that contain potential threats (such as queries with UNION SELECT, DROP, OR OR 1=1) to prevent attackers from obtaining internal database information through SQL injection. Denial of Service (DoS) attacks are also the focus of database firewall protection. In this attack, the attacker depletes the database's resources by sending a large number of requests, making it unable to respond to requests from legitimate users. The database firewall prevents DoS attacks by limiting the number of requests per unit time and setting the maximum number of connections to ensure the stability and availability of the database.

In the face of these attacks, the firewall can also use rule and policy management to filter database requests through preset rule sets and policies. Administrators can formulate detailed access control rules based on the IP address of the visitor, the source of the request, the query content, and the user role. Firewalls can be configured with rules that restrict certain users or IP segments from accessing the database only during certain periods of time, or that restrict them to performing only read operations. The database firewall also has the ability to monitor and analyze database traffic in real time. Such as a sudden large number of database connection requests, query execution time beyond the normal range, or users trying to access data that does not match their roles, these abnormal activities are usually the precursor of database attacks or intrusions. Firewalls can learn and recognize normal database access patterns. When abnormal behavior is detected, an alarm is triggered or access requests are directly blocked to prevent potential threats from spreading further.

The database firewall logs all database access behavior, including approved or denied requests, unusual behavior, and potential security threats. These logs provide the basis for subsequent security audits. When a database is attacked or a data breach occurs, the administrator can analyze logs to trace the source of the attack and take appropriate measures. The audit function of the firewall not only helps locate the firewall quickly, but also optimizes the rule configuration to improve the protection effect.

5. Conclusion

Under the large-scale concurrent data access, the traditional database connection mode faces the performance bottleneck and other problems. By introducing the connection pool technology, distributed database optimization, and cache mechanism, the database access efficiency can be effectively improved, the connection cost can be reduced, and the high availability of the system can be ensured. The security management of databases has also become more complex and critical. Technical measures such as identity authentication and permission control, multi-level data encryption, log audit and database firewall can effectively resist potential security threats and ensure data integrity, confidentiality and availability. In the future, with the further development of big data technology, the optimization of database connection and access technology needs to be more intelligent and automated, and security management also needs to pay more attention to fine-grained control and dynamic defense mechanisms to cope with the complex and ever-changing network environment and increasingly severe security challenges.

References

[1] Chengzhi Wang. Analysis of computer database connection access technology based on Big Data [J]. Software, 2022, (6): 43.
[2] Benhan Zhang. Analysis and research on computer database connection and access technology in the era of Big Data [J]. Software, 2023, 44(5):154-156. (in Chinese)
[3] Y Fan. Analysis and research on computer database connection and access technology in the era of big data [J]. Digital technology and applications, 2020, 38 (5): 2. DOI: 10.19695 / j.carol carroll nki cn12-1369.2020.05.67.
[4] Zehua Qiao. Research on security management Technology based on computer network database [J]. Information Recording Materials, 2022, (1): 23.
[5] Xing Wang. Analysis of database programming technology based on computer Software engineering [J]. Hunan Paper Making, 2022, (1): 51.
[6] Y Yin. Analysis of computer database connection access technology based on big data [J]. Information and computers, 2021, 33 (1): 2. DOI: 10.3969 / j.i SSN. 1003-9767.2021.01.054.